UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/784,374	02/23/2004	Pal Takacsi-Nagy	BEAS-01389US2	8929

23910 7590 01/16/2008
FLIESLER MEYER LLP
650 CALIFORNIA STREET
14TH FLOOR
SAN FRANCISCO, CA 94108

EXAMINER	
WANG, JUE S	
ART UNIT	PAPER NUMBER
2193	
MAIL DATE	DELIVERY MODE
01/16/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/784,374	TAKACSI-NAGY ET AL.	
	Examiner	Art Unit	
	Jue S. Wang	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 October 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-13,15-30,32-37 and 40-42* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-13,15-30,32-37 and 40-42* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *10/29/2007*.	6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-13, 15-30, 32-37, and 40-41 have been examined.

2. Claims 14, 31, 38, and 39 were cancelled in Amendment dated 10/29/2007.

Specification

3. The specification is objected to under 35 CFR 1.75 because there is no description for the "computer readable medium" recited in claim 40. It should be known that "air", "wireless transmission", and the like are computer readable media, but they are non-statutory claim subject matters. Accordingly, the specification fails to limit "computer readable medium" to embodiments which fall within a statutory category.

Double Patenting

4. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

5. Claims 1-5, 18-22, and 35-37 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 50, 52, 57, 59, 64, and 66 of copending application 10/784375.

6. Although the conflicting claims are not identical, they are not patentably distinct from each other because both the present application and copending application 10/784375 describe a programming language extended with XML constructs for defining parallel processing. For example, claim 1 of the present application discloses extending an existing language by adding at least one language construct defined by a second language, and claims 52 of the copending application 10/784375 discloses a Java programming language extended with a plurality of workflow constructions provided as XML commands.

7. These are provisional obviousness-type double patenting rejections because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

9. Claims 1-3, 18-20, and 35-37 are rejected under 35 U.S.C. 102(a) as being anticipated by Christensen et al., "Extending Java for High-Level Web Service Construction" (hereinafter Christensen).

10. As per claim 1, Christensen teaches the invention as claimed, including a method for extending an existing programming language, comprising the steps of:

selecting an existing programming language; and,

extending an existing programming language by adding at least one language construct defined by a second language (see page 6, paragraphs 3, 6, page 7, Fig 3, page 9; EN: element tags are XML constructs added to Java).

11. As per claim 2, Christensen further teaches that the existing programming language is an object oriented language (see page 6, paragraph 3, page 7, Fig 3).

12. As per claim 3, Christensen further teaches that the second language is XML (see page 6, paragraph 3, page 9).

13. As per claims 18-20, these are the computer system claims of claims 1-3. Therefore, they are rejected using the same reasons as claims 1-3.

14. As per claim 35, this is the system claim of claim 1. Therefore, it is rejected using the same reasons as claim 1.

15. As per claim 36, Christensen teaches the invention as claimed, including a method for extending object oriented programming language, comprising the steps of:

selecting object oriented programming language; and, extending the object oriented programming language by adding at least one language construct defined by XML (see page 6, paragraphs 3, 6, page 7, Fig 3, page 9; EN: element tags are XML constructs added to Java).

16. As per claim 37, this is the storage medium claim of claim 36. Therefore, it is rejected using the same reasons as claim 36.

Claim Rejections - 35 USC § 103

17. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

18. Claims 4-6, 12, 13, 21-23, and 29-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Christensen et al., "Extending Java for High-Level Web Service Construction" (hereinafter Christensen), in view of Meredith, (US 6,516,322 B1).

19. As per claim 4, Christensen does not teach a parallelism construct representing parallel branch of program execution.

Meredith teaches a scheduling language SLANG which is written in XML, and includes syntax that allows for expression of features associated with the model for describing distributed and parallel computing (see column 2, lines 14-20, column 6, lines 26-38, and column 12, lines

45-64), where SLANG has a parallelism construct representing parallel branch of program execution (the task construct, see Figs 16b-16d, column 13, lines 3-5, 46-54, and column 14, lines 57-67).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen such that the XML integrated into Java contains construct to specify workflow including a parallel construct representing parallel branch of program execution as taught by Meredith because the approach of Christensen generalizes in a straightforward manner to any arbitrary interaction language described by an XML schema (see page 3, paragraph 4 of Christensen) and the XML constructs provide compositional specification of autonomously executing systems such that the cost of developing and managing applications that span business units connected by communications networks is greatly reduced (see column 6, lines 26-55 of Meredith).

20. As per claim 5, Christensen does not teach that the parallelism construct further comprises a plurality of branch constructs defined by said second language, wherein said branch constructs represent parallel branches of program execution comprising of at least one software activity.

Meredith teaches that the parallelism construct further comprises a plurality of branch constructs defined by said second language, wherein said branch constructs represent parallel branches of program execution comprising of at least one software activity (the action construct, see column 13, lines 3-5, 46-55, column 14, lines 16-24, 57-67).

21. As per claim 6, Christensen does not teach said parallelism construct is further nested within a similar parallelism construct.

Meredith teaches that the parallelism construct is further nested within a similar parallelism construct (the task construct is nested within the partition construct which is also a parallelism construct, see Figs 6, 16b-16d, 23b, column 14, lines 57-67, column 15, lines 54-61).

22. As per claim 12, Christensen does not teach that said language construct is an action construct representing an activity that allows a first software component written using the extended existing programming language to call an operation on a second software component written using said existing programming language.

Meredith teaches that the language SLANG has the functionality a first software component written using the programming language XML to call an operation on a second software component written using another programming language such as Java (see column 13, lines 3-5, 46-55; specifically "the action can be mapped to invocations on common object model (COM) objects, … or other native technology behavior"). While Meredith does not specifically teach a construct to do the mapping, it would have been obvious to one of ordinary skill in the art at the time of the invention to extend SLANG with such a construct to denote the mapping since XML itself is an extensible language. Furthermore, while Meredith does not explicitly teach that the second software component is written in Java, it would have been obvious that the native technology behavior includes software components written in Java since Java is a popular programming language with the advantage of being platform independent.

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen such that the XML integrated into Java contains constructs to specify workflow including an action construct representing an activity that allows a first software component written using the extended programming language (XML) to call an operation on a second software component written using another programming language as taught by Meredith because the approach of Christensen generalizes in a straightforward manner to any arbitrary interaction language described by an XML schema (see page 3, paragraph 4 of Christensen) and the XML constructs provide compositional specification of autonomously executing systems such that the cost of developing and managing applications that span business units connected by communications networks is greatly reduced (see column 6, lines 26-55 of Meredith).

23. As per claim 13, Meredith teaches that the action constructs allows said first software component to call a piece of object oriented programming language code (i.e., "the action can be mapped to invocations on common object model (COM) … or other native technology behavior", see column 13, lines 3-5, 46-55).

24. As per claims 21-23 and 29-30, theses are the computer system claims of claims 4-6 and 12-13. Therefore, they are rejected using the same reasons as claim 4-6 and 12-13.

25. Claims 7, 8, 11, 24, 25, and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Christensen et al., "Extending Java for High-Level Web Service Construction" (hereinafter Christensen), in view of Mital et al. (US 7,184,867 B1, hereinafter Mital).

26. As per claim 7, Christensen does not teach that said language construct is a transaction construct representing transaction block of at least one software activity.

Mital teaches a tool to allow a user to create a schedule for business workflow where the schedule can be converted to a scheduling programming language written in XML (SLANG), where the model allows the user to define transactions representing transaction block of at least one software activity which are translated to the context constructs (see Fig 2, Fig 3, Fig 24, column 2, lines 29-46, column 9, lines 47-60, and column 16, lines 40-65).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen such that the XML integrated into Java contains constructs to specify workflow including a transaction construct representing transaction block of at least one software activity as taught by Mital because the approach of Christensen generalizes in a straightforward manner to any arbitrary interaction language described by an XML schema (see page 3, paragraph 4 of Christensen) and the XML constructs are used to specify workflow which facilitates the processing of business transactions between different companies (see column 1, line37 – column 2, line 46 of Mital).

27. As per claim 8, Christensen does not teach that the transaction construct further specifies the number of retry attempts to perform the software activities inside said transaction block.

Mital that the transaction construct of SLANG further specifies the number of retry attempts to perform the software activities inside said transaction block (see column 2, lines 29-46, column 9, lines 47-60, column 13, lines 8-9 and Table 2, and column 16, lines 40-65; EN: while Mital does not explicitly state as such, it would have been obvious that the transaction construct specifies the number of retry attempts since retry count is one of the properties of the transaction shape and the model is converted to SLANG).

28. As per claim 11, Christensen does not teach said language construct is an exception handler construct representing an execution mechanism comprising of exception handler construct defined by said second language, which represents exception not caught by the programming language handler methods.

Mital teaches a tool to allow a user to create a schedule for business workflow where the schedule can be converted to a scheduling programming language written in XML (SLANG), where the model allows the user to define an exception handler construct representing an execution mechanism comprising of exception handler construct (see column 2, lines 29-46, column 9, lines 47-60, and column 16, lines 40-65; EN: while Mital does not explicitly state that SLANG has an exception handler, it would have been obvious that SLANG has an exception handler since the transaction shapes have catch pages associated to define routines invoked on a failed transaction, and the model is converted to SLANG).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen such that the XML integrated into Java contains constructs to specify workflow including an exception handler construct representing an execution mechanism comprising of exception handler construct (EN: the XML construct would not be caught by Java since it's defined in XML) as taught by Mital because the approach of Christensen generalizes in a straightforward manner to any arbitrary interaction language described by an XML schema (see page 3, paragraph 4 of Christensen) and the XML constructs are used to specify workflow which facilitates the processing of business transactions between different companies (see column 1, line37 – column 2, line 46 of Mital).

29. As per claims 24, 25, and 28, these are the computer system claims of claims 7, 8, and 11. Therefore, they are rejected using the same reasons as claims 7, 8, and 11.

30. Claims 9, 10, 26, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Christensen et al., "Extending Java for High-Level Web Service Construction" (hereinafter Christensen), in view of Mital et al. (US 7,184,867 B1, hereinafter Mital), as applied to claims 7 and 24 above, further in view of Alonso et al. "Advanced Transaction Models in Workflow Contexts", (hereinafter Alonso).

31. As per claim 9, Christensen and Mital do not teach that said transaction construct is further enclosed within a saga construct comprising a compensation construct with at least one compensating software activity to undo work associated with the transaction block if the transaction block is aborted, wherein the saga construct represents a long running transaction.

Alonso teaches a method to convert high level specifications of advanced transaction models such as linear sagas for long lived transactions with compensation activity into workflow processes, where the compensation active undoes work associated with the transaction block if the transaction block is aborted (see pages 4-5, section 4, paragraphs 1, 2, and section 4.1).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen and Mital to implement linear sagas using the preexisting XML constructs used to define workflow including compensations for transactions (see column 9, lines 59-63, column 50, lines 50-56) as taught by Alonso because linear sagas provide a model to deal with long lived transactions and allow a transaction to release resources before committing (see page 4, right column, last paragraph of Alonso). Furthermore, while Christensen, Mital, and Alonso do not teach a saga construct to denote the saga, it would have been obvious to one of ordinary skill in the art at the time of the invention to extend SLANG with such a construct to mark the saga functionality implemented since XML itself is an extensible language.

32. As per claim 10, Christensen and Mital do not teach that said saga construct further comprises a plurality of transaction blocks. However, Alonso teaches that the linear saga comprises a plurality of transaction blocks (see page 6, Figure 1), so the saga construct of Christensen modified by Mital and Alonso would have a plurality of transaction blocks.

33. As per claims 26 and 27, these are the system claims of claims 9 and 10. Therefore, they are rejected using the same reasons as claims 9 and 10.

34. Claims 15-17 and 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Christensen et al., "Extending Java for High-Level Web Service Construction" (hereinafter

Christensen), in view of van der Aalst et al., "XML Based Schema Definition for Inter-Organization Workflow", (hereinafter Aalst).

35. As per claim 15, Christensen does not teach that said language construct is a multiple receive construct that allows a software component written using the extended existing programming language to wait on multiple input events received.

Aalst teaches the language XRL which is based on XML syntax and provides support for routing of workflow among trading partners for internet based electronic commerce services (see page 1, section 1, paragraph 1), where XRL has a multiple receive construct that allows a software component to wait on multiple input events received (the wait_any construct, see pages 16-17, section 4.7).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen such that the XML integrated into Java contains constructs to specify workflow including a multiple receive construct that allows a software component written using the extended existing programming language to wait on multiple input events received as taught by Aalst because the approach of Christensen generalizes in a straightforward manner to any arbitrary interaction language described by an XML schema (see page 3, paragraph 4 of Christensen) and the XML constructs provide support for routing of workflow among trading partners to Internet based electronic commerce services to facilitate increased productivity and interoperability (see page 1, section 1, paragraph 1 of Aalst).

36. As per claim 16, Christensen does not teach that said multiple receive construct further allows said software component proceed on a particular branch of program execution, based on the input event that occurred first within the said multiple input events.

While Aalst does not teach that the multiple receive construct allows software component proceed on a particular branch of program execution, based on the input event that occurred first within the said multiple input events. However, it would have been obvious that this functionality can be achieved in conjunction with the condition construct (see page 16, section 4.6, page 24, paragraph 2, and page 26, Figure 12) by placing the condition construct directly after the wait_any construct such that different branches are taken depending on the event that was received by wait_any. Furthermore, while Christensen and Aalst do not teach the multiple receive construct has the specified functionality, it would have been obvious to one of ordinary skill in the art at the time of the invention to augment the multiple receive construct of XRL with such a functionality implemented since XML itself is an extensible language.

37. As per claim 17, Christensen does not teach that said construct is a looping construct with ordering of messages received, representing looping functionality, wherein the order allows said messages to be received in an order.

Aalst teaches the language XRL which is based on XML syntax and provides support for routing of workflow among trading partners for internet based electronic commerce services (see page 1, section 1, paragraph 1), where XRL has a looping construct representing looping functionality (see pages 16-17, section 4.7, the while_do construct). While Aalst does not specifically teach that the loop construct has an ordering of messages received, where the ordering allows said messages to be received in an order, it would have been obvious that an ordering of the messages received does not need to rely on a looping construct, and instead can be achieved from the content of the loop. For example, in the mail order processing example presented on pages 24 –27, section 6, the while_do loop is used to find a shipper where a list of shippers is contacted in order to find an available shipper. Since a message is received to determine whether the shipper can perform the shipment and the shippers are contacted in order, the messages are received in order because the shippers are contacted in order according to a list (see page 24, paragraph 2). Furthermore, while Aalst does not teach the loop construct has the specified functionality, it would have been obvious to one of ordinary skill in the art at the time of the invention to augment the loop of XRL with such a functionality since XML itself is an extensible language.

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Christensen such that the XML integrated into Java contains constructs to specify workflow including a looping construct with ordering of messages received, representing looping functionality, wherein the order allows said messages to be received in an order as taught by Aalst because the approach of Christensen generalizes in a straightforward manner to any arbitrary interaction language described by an XML schema (see page 3, paragraph 4 of Christensen) and the XML constructs provide support for routing of workflow among trading partners to Internet based electronic commerce services to facilitate increased productivity and interoperability (see page 1, section 1, paragraph 1 of Aalst).

38. As per claims 32-34, these are the computer system claims of claims 15-17. Therefore, they are rejected using the same reasons as claims 15-17.

39. Claims 40-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over van der Aalst et al., "XML Based Schema Definition for Inter-Organization Workflow", (hereinafter Aalst), in view of Christensen et al., "Extending Java for High-Level Web Service Construction" (hereinafter Christensen).

40. As per claim 40, Aalst teaches the invention as claimed, including a system for utilizing a workflow language, comprising:

a computer including a processing device operating thereon (see page 2, paragraph 2, page 10, last paragraph);

a source file stored on a computer readable medium (i.e., an XRL file, see page 10, last paragraph, page 11, Figure 4), wherein the source file includes a workflow definition created using a workflow language (see page 11, page 12), wherein said workflow language comprises a plurality of workflow constructs defined by XML (see pages 12-17, section 4), including constructs for defining loop processing that performs a plurality of activities until a condition attribute is true (i.e., the While_do construct, see page 13, item 11, page 16, section 4.6, pages 24-25); and

means for creating a workflow program according to said workflow definition including

means for the computer to read the source file and process the plurality of activities (see page 10, last paragraph, page 11, Figure 4), means for determining whether the condition attribute is true (see page 16, section 4.6),

means for performing the plurality of activities when the condition attribute is not true (see page 16, section 4.6), and

means for terminating the loop processing when the condition attribute is true (page 16, section 4.6).

Aalst teaches the plurality of activities are performed until the condition evaluates to true, it would have been obvious that one of ordinary skill in the art who is capable of designing and implementing programming languages could have provided the looping construct with a different semantics such that the activities are performed so long as a condition is true so that the plurality of activities are performed when the condition is true and the loop processing is terminated when the condition is not true because this semantics of while loops are more commonly found in existing programming languages and is therefore more familiar to programmers.

Aalst does not teach that the workflow language comprises an object oriented programming language extended with the plurality of constructs defined by XML.

Christensen teaches extending the Java programming language with XML constructs (see page 6, paragraph 3, page 7, Fig 3, page 9).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Aalst such that the workflow language comprises an object oriented programming language extended with the plurality of constructs defined by XML using the extension technique taught by Christensen because the approach generalizes in a straightforward manner to an arbitrary interaction language described by an XML schema and the Java language offers lots of built-in features for developing modern web services such as network support and strong network guarantees (see page 1, paragraph 1, page 3, paragraph 3 of Christensen).

41. As per claim 41, Aalst further teaches the means for determining whether the condition attribute is true, determines whether the condition is true before the plurality of activities are performed (see page 16, section 4.6).

42. As per claim 42, Aalst further teaches the means for determining whether the condition attribute is true, determines whether the condition is true after the plurality of activities are performed (see page 16, section 4.6).

Response to Arguments

43. <u>Rejection of claims 1-37 U.S.C. §102(a) and §103(a)</u>:

44. Applicants' arguments with respect to claims 1-37 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

45. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- o Benton et al., "Interlanguage Working Without Tears: Blending SML with Java", 1999, ACM SIGPLAN Notices, volume 34, issue 9, pages: 126-137.

  This document is cited to teach extending the SML language with Java constructs.

46. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jue S. Wang whose telephone number is (571) 270-1655. The examiner can normally be reached on M-Th 7:30 am - 5:00pm (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 571-272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Jue Wang
Examiner
Art Unit 2193

[illegible]
MENG-AI T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100